Abstract

The objective of the research work documented in the thesis titled “Power Aware Software” is to introduce new compilation techniques for reducing dynamic, switching and leakage power on modern processors. The works “Energy Efficient Software Prefetching using Dynamic Voltage Scaling with Prefetch Distance Scaling” and “Energy Efficient Multiway Branch Translation Techniques” save dynamic power dissipation. The work “Loop Unrolling with Partial Gray Code Sequence” minimizes energy consumption due to bus switching activity. The work “Loop Unrolling with Fine Grained Power Gating” saves runtime leakage power of unused functional units. The work “Energy Optimization Techniques for OpenMP” does joint optimization of dynamic and leakage power saving for parallel programs using OpenMP which run on multicore processors. Each of the work is discussed briefly in the following paragraphs.

The work “Energy Efficient Software Prefetching using Dynamic Voltage Scaling with Prefetch Distance Scaling” presents a mechanism to transform a program with software prefetching to its power-aware equivalent. This is done by executing the software prefetching program with different voltage-frequency pairs. Besides reducing the power, the performance is improved by adjusting the prefetch distance. Experimental results of the proposed scheme guarantees that performance improvement of software prefetching program is possible at the cost of lesser power consumption.

The work “Energy Efficient Multiway Branch Translation Techniques” introduces the use of k-d tree and pattern matcher to generate efficient code, i.e. lesser execution time, for multiway branch. However, instead of enhancing performance, Voltage Frequency Scaling (VFS) is applied to achieve energy efficiency without degradation in performance. The proposed work is evaluated on a wide range of benchmark programs. The BTB energy saving lies in the range of 20% to 80% with small improvement in performance as well. The total energy reduction is in the range of 3-12%.

“Loop Unrolling with Partial Gray Code Sequence” is a software technique that reduces switching activity as well as energy consumption on the address bus of on-chip data memory, which is independent of process technology parameters. Loop unrolling with partial Gray code sequence is suitable for array computations where there are sequential access of array elements, which allows to reschedule the access of array elements in a Gray coded sequence of their addresses, in each iteration of the unrolled loop. The proposed scheme is only applicable to the Harvard Architecture. The expressions for switching activity and energy consumed on the address bus of the on-chip data memory are derived for both unrolled loop with and without partial Gray code sequence. The proposed translation method finds a relocatable base address of the array so that the partial Gray code sequence is maintained without any energy-performance overhead and achieves a considerable amount of energy reduction without any performance loss. Array unification is introduced for multiple arrays taking part in computation within a loop. An algorithm that performs array unification and translation of the loop with multiple arrays to its loop unrolled version with partial Gray code sequence has been proposed. The efficacy of the proposed approach is evaluated on five sample programs and ten benchmark programs. 10–93%

reduction in switching activity and 10–94% reduction in energy dissipated on the address bus of on-chip data memory have been achieved.

“Loop Unrolling with Fine Grained Power Gating” introduces a compilation technique to reduce runtime leakage power of functional units of a processor by combining loop unrolling with power gating. The instructions in the unrolled loop are scheduled to provide opportunities for power gating the functional units which are not used for a considerable amount of time. An algorithm that maximizes the savings of leakage energy without performance loss due to execution of power gating instructions has been introduced. The algorithm involves loop unrolling, scheduling of instructions and finally insert power gating instructions. The present work is explained using two illustrative examples, one without loop-carried dependence and the other one with loop-carried dependence. It is observed that the number of clock cycles taken by the power gating instructions is less than or equal to the number of clock cycles saved by loop unrolling. This results in 23–64% reduction of the total energy consumed by the benchmark programs without any degradation of performance.

The work “Energy Optimization Techniques for OpenMP” introduces energy efficient scheduling of OpenMP parallel loop on multi-core processors having fine-grained dynamic voltage frequency scaling (DVFS) and dynamic threshold voltage scaling (DVTS) facility. Slack iterations in an OpenMP loop schedule are partitioned among the cores to obtain an energy efficient loop schedule. The slack in iterations are partitioned among the cores allow us to apply DVFS and DVTS, to reduce dynamic and leakage energy consumption. Considering the time taken by the original OpenMP loop schedule as deadline, the clock frequency and supply voltage of the cores are scaled down, and the threshold voltage of the cores are scaled up. A constant time loop scheduling algorithm has been proposed, which guarantees a loop schedule consuming minimum energy for a given OpenMP loop schedule. The energy efficient loop schedules are obtained for static, dynamic, guided, runtime and auto OpenMP loop scheduling algorithms. The energy efficient versions of work sharing and critical section constructs in OpenMP are also introduced. The proposed approach is tested on sample as well as benchmark programs having parallel and critical regions. The combination of DVFS and DVTS causes substantial reduction in dynamic and leakage energy with negligible delay and energy overhead.

The objective of our work was to develop compilation techniques for energy efficient code generation to reduce dynamic and leakage power consumption in modern processors. In the process of this exercise, we came up with several novel solutions to the challenges of reducing dynamic and leakage power without degradation in performance. According to our understanding and study, we believe that this work is helpful in designing energy efficient power aware softwares. In future, the compiler designers as well as programmers designing energy efficient and power aware softwares can utilize the software techniques proposed in this thesis.

Keywords – Dyamic power, switching activity, leakage power, power aware, energy efficient, dynamic voltage and frequency scaling, software prefetching, prefetch distance scaling, loop unrolling, Gray code sequence, array unification, fine-grained, power gating, loop scheduling, slack partitioning, dynamic threshold voltage scaling, synchronization constructs.